Subject: Diploma project

Supervisor: Josefin Reimark

Programming Languages: Energy Consumption and Role in Green Computing

Michael Jiacheng Yu

Kungsholmens Gymnasium

NA20EC

Michael Yu NA20EC

Abstract

The aim of this study was to compare the energy consumption between programming languages and determine whether the choice of programming language had a significance in improving energy efficiency when performing large computations. Four programming languages, namely C, C++, JavaScript, and Python, were examined in terms of runtime and energy consumption. To evaluate their performance, three distinct programming problems were implemented in each language, ensuring identical time and space complexity. The results show that the faster the programming language, the more energy efficient it was, with C and C++ demonstrating the highest efficiency, followed by JavaScript and Python. In terms of wattage, JavaScript and Python were found to be the most efficient, followed by C++ and C. The findings suggest that opting for the most energy-efficient programming language can potentially lead to reduced greenhouse gas emissions. This approach of picking programming language has the potential to be a new environmentally responsible practice in writing code.

Table of Contents

1. Introduction

In 2021, the global electricity consumption of data centers represented 0.9-1.3% of global electricity consumption and is expected to continue growing (IEA, 2022). The technologies driving the demand for processing include artificial intelligence, cloud computing, and blockchain: recent advances in technology that have greatly contributed to the surge in energy consumption related to the information and communications technology sector. (IBM, 2023). In artificial intelligence alone, the amount of data processing is expected to grow exponentially, with a 3.4-month doubling time (Amodei & Hernandez, 2018). Such increase in processing, particularly with respect to big data and data mining, has emphasized the industry standards of mitigating energy consumption associated with software by optimizing computational efficiency through the use of efficient algorithms. This approach is commonly referred to as green coding and has gained popularity in recent years (IBM, 2023). However, in addition to algorithm optimization, the programming language itself can also have a significant impact on the computational efficiency of a piece of code, due to inherent differences in how they are processed by a computer (Fourment & Gillings, 2008). Therefore, with the question of energy efficiency in code processing becoming increasingly relevant, it is of interest to investigate how programming languages compare in terms of energy consumption when performing large computations.

1.1 Aim

To compare the energy consumption between programming languages and determine whether the choice of programming language has a significance in improving energy efficiency when performing large computations.

1.2 Research Questions

RQ1: How does the energy consumption of processing differ across different programming languages when performing the same set of tasks?

RQ2: Is the faster programming language always the most energy efficient?

RQ3: How can the choice of programming language lower the environmental impact of big data processing and data mining?

2. Theory

2.1 Code Processing and Programming Languages

Computer programs are made up of a list of unambiguous instructions that can be run mechanically by a processor hardware, known as the Central Processing Unit (CPU). At the hardware level, the CPU is only able to perform instructions written in an elementary language called machine language, that is expressed in binary (Eck., 2014). Instead of writing instructions in binary, a high-level language, also commonly referred to as a programming language, can be used to write computer instructions in a more human-readable syntax, like English (Portland State University, n.d.). Therefore, when code is written in a high-level language, it must first be translated into machine code in order to be run by the CPU. Depending on the translation method for a high-level language, it can be categorized as a compiled or interpreted language. The higher level a programming language is, the higher the level of abstraction is.

Compiled languages are high-level languages that use a program called a compiler to translate the source code into machine code. This process generates a binary file that can be run directly, without any reference to the source code. C and c++ are examples of compiled languages.

On the other hand, interpreted languages are high-level languages, usually of a higher level than compiled languages, that use a program called an interpreter to run the source code. Unlike compiled languages, an interpreter reads and translates the code line-by-line while the program is being run. As a result, the source code must be interpreted every time the program runs. JavaScript and Python are examples of interpreted languages.

While interpreted languages are easier to debug and revise, they are generally slower than compiled languages due to the need for translation during runtime.

According to a survey conducted by Slash Data with a sample size of 26,000 respondents, the top five programming languages are ranked in the following order: JavaScript, Python, Java, C, and C++ (Fanouraki, 2022).

2.2 Energy Measurement of CPUs

Energy consumption of a computer system can be measured in several ways. One of the methods is to use Intel's energy measurement feature called the Running Average Power Limit (RAPL). RAPL is a feature of Intel CPUs that enables software to measure energy consumption of different components, such as the CPU and the memory, in real-time.

RAPL works by utilizing internal counters and sensors integrated within the CPU to record the energy usage of components at regular intervals, providing a continuous stream of energy-related data. By integrating energy values over time, RAPL can calculate the energy consumed by a particular component for a specific duration. This energy measurement is typically represented in units like joules or watt-hours. (Intel, 2022)

2.3 Algorithm Efficiency

In the analysis of algorithms, time complexity and space complexity are two concepts used to measure efficiency. Time complexity refers to the amount of time an algorithm takes to run as a function of the size of its input. It measures how many operations an algorithm performs in the worst-case scenario. The time complexity is often expressed using big O notation, where O(f(n)) denotes an upper bound on the number of operations an algorithm will perform, with n representing the input size and f(n) being a function that describes the number of operations required. Space complexity, on the other hand, refers to the amount of memory space an algorithm uses to run as a function of the size of its input. It measures how much memory an algorithm requires in the worst-case scenario. Space complexity is also usually expressed in terms of big O notation, where O(f(n)) denotes an upper bound on the amount of memory space an algorithm will require, with n representing the input size and f(n) being a function that describes the memory required. For instance, an algorithm with time complexity O(n^2), would take approximately n^2 operations to execute on an input of size n (see figure 1). Therefore, algorithms with the same time and space complexity will be equally efficient in terms of computational complexity; theoretical performance.

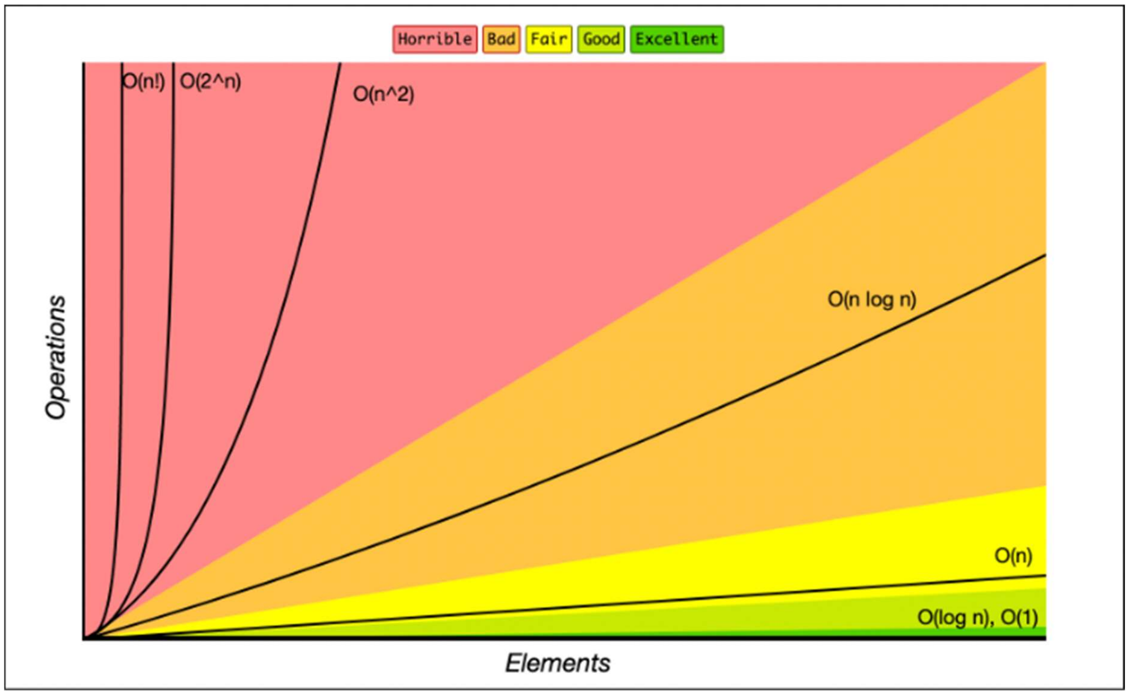

Figure 1. Comparison of Big O Complexities for different operations as a function of elements (Rowell, 2016)**.**

2.4 Sustainability

The Sustainable Development Goals (SDGs) are a set of 17 global goals established by the United Nations in 2015. They aim to address global social, economic, and environmental challenges facing the world today and act as a blueprint to achieve a more sustainable future. Both the development of industry as well as the improvement in energy efficiency within industry are highlighted as crucial in reaching several of these goals. Improving energy efficiency in large-scale data processing connects to both environmental and economical sustainability, specifically seen in SDGs 7, 9 and 13. SDG 7 calls for the efficient, clean, and predictable usage of energy to ensure reliable and sustainable energy for all, with its subgoal 7.3 specifically relating to the global improvement in energy efficiency. SDG 9 mentions the importance of retrofitting industries with focus on the adoption of clean and environmentally sound technologies for sustainable industry. With the current industry standard for improving energy efficiency in processing being green coding, which solely focuses on algorithm efficiency, the choice of programming language could become a new complementary industry standard if it has a noticeable impact on energy efficiency in large-scale processing. Furthermore, SDG 13 intends to combat climate change. (United Nations, n.d.) In 2020, data centers accounted for around 300 $MtCO_2$-eq, or nearly 1% of global energy-related green house gas emissions and is projected to increase to 2% by the end of 2023 (IEA, 2022). The improvement in energy efficiency within large-scale data processing is therefore relevant in achieving this goal.

3. Method

Set up of testing environment

A desktop machine, with specifications listed in table 1, was set up and installed with compilers for C, C++, and interpreters for JavaScript and Python. The version of each compiler and interpreter was picked by their latest stable release as of January 20, 2023 (see table 2).

Table 1. Specifications of the testing machine used for measuring energy and time consumption.

Operating System	Ubuntu Desktop 22.04.2 LTS running Linux 5.15.103
CPU	Intel Core i5-13600k CPU at 3.50 GHz
RAM	16GB of DDR4 RAM at 3600 MHz
Motherboard	Gigabyte B660 AORUS Master DDR4
GPU	NVIDIA GeForce RTX 3070

Table 2. Release version of each programming language used in the measuring of energy and time consumption.

Programming Language	**Version**
C	gcc 11.3.0
C++	g++ 11.3.0
JavaScript	node 12.22.9
Python	python 3.10.6

The machine was then set up with an energy measuring framework[1] that utilized Intel's RAPL interface, in the form of a python library called pyRAPl 0.2.3.1, to measure energy and time consumption in microjoules and microseconds respectively (Intel, 2022). The RAPL interface was selected due to its ability to accurately measure CPU energy usage and since all code was processed exclusively in the CPU. The energy measuring framework included the pyRAPL library in the Brain.py python code and contained each benchmark test (see table 3), along with their respective input data and language-specific solutions. Notably, all benchmark tests chosen were able to be solved in constant space complexity, and each benchmark solution used had the same space and time complexity across all programming languages, with space complexity being constant to negate the case of performance limitations due to excessive memory usage. The overall workflow of the energy measuring process is as described in figure 2.

Table 3. Benchmarks used in testing.

Benchmark	**Description**	**Time Complexity**	**Input Size (Million)**
Loop	While loop that increments by 1 until the input size.	O(n)	1000
Linear Regression	Linear regression analysis between two variables, each containing data points half the input size.	O(n)	20
Heap Sort	Sorts data points of input size using the heap sort algorithm.	O(n log(n))	10

[1]The measuring framework and results are publicly available at www.github.com/MichaelWhyYou/diploma-project.

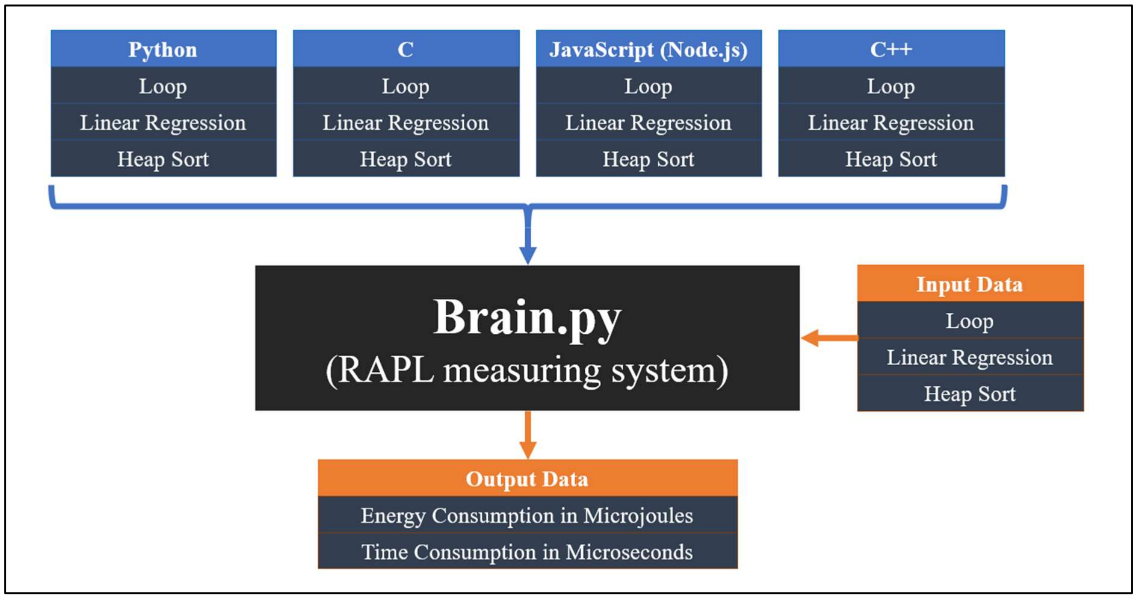

Figure 2. Workflow of the energy measuring framework.

3.2 Lab Procedure

The desktop machine was started, and the Linux operating system was logged in to. To ensure no background updates taking place, which could in turn affect CPU-performance, Wi-Fi was turned off, and 10 minutes were waited to let the machine reach idle performance. Then the following procedure was followed:

1. The Brain.py program was started, and the C programming language was selected when prompted. Ten trials of each benchmark were then run uninterruptedly, and the energy and time consumption data outputted by Brain.py was recorded for each trial, and the average energy and time taken for each benchmark calculated.
2. 10 minutes were then waited for the machine to reach idle performance.
3. The above steps (1-2) were repeated for each programming language, selecting a different language each time in the prompt.

4. Results

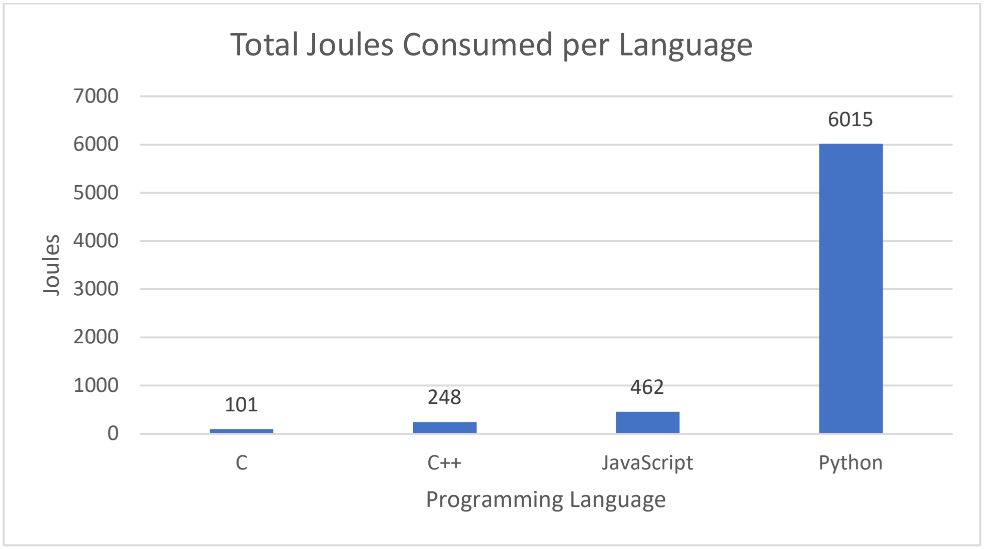

Figure 3. Energy per programming language for executing the benchmark.

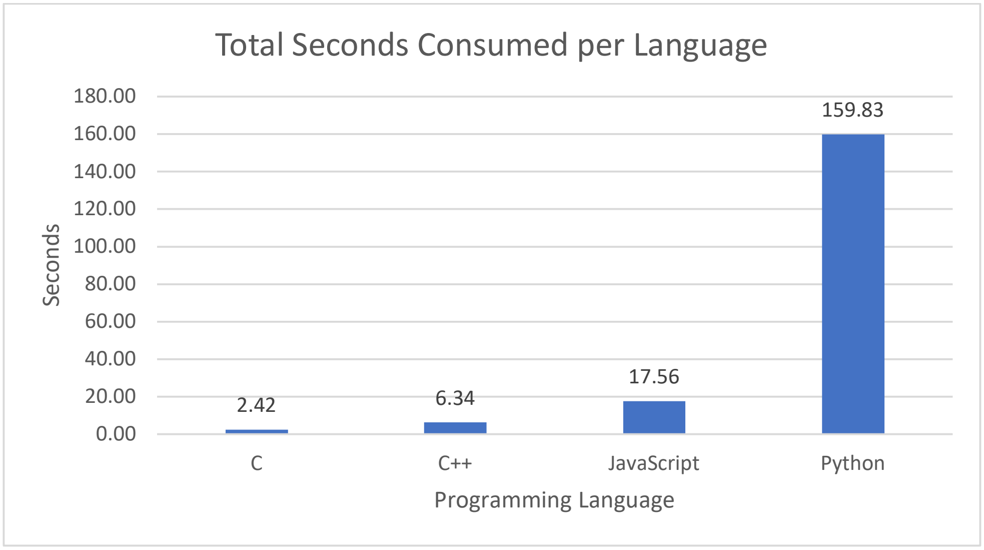

Figure 4. Time per programming language for executing the benchmark.

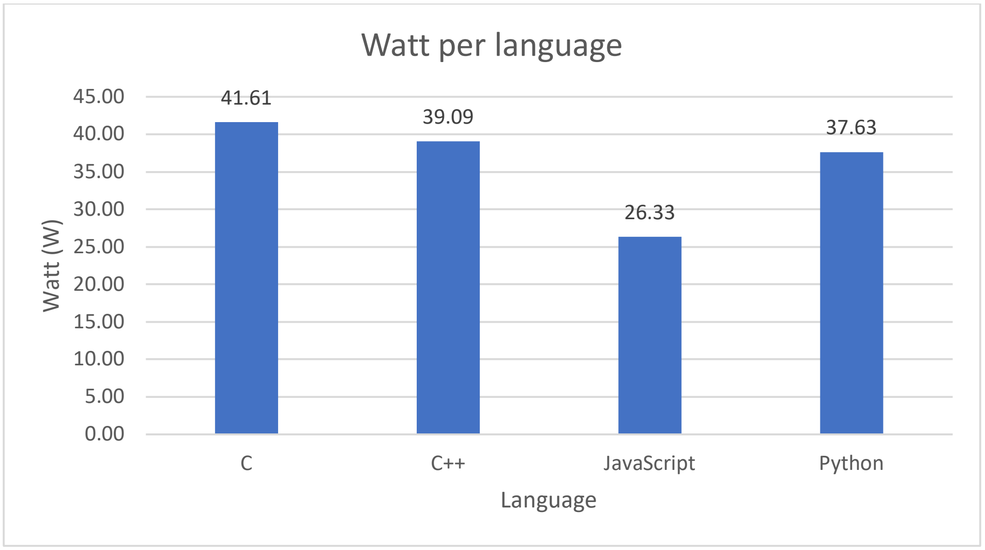

Figure 5. Watt per programming language for executing the benchmark.

5. Discussion

RQ1: How does the energy consumption of processing differ across different programming languages when performing the same set of tasks?

The total energy consumption of processing between the programming languages tested differed when running the same benchmark (see figure 3). The languages were observed to consume energy in the following order, from least to most energy consumed: C, C++, JavaScript, and Python. The largest difference in energy consumption was between C and Python, with Python consuming around 60 times more energy than C. It is worth noting that C and C++ are compiled languages, while JavaScript and Python are interpreted, and that in this experiment, both C and C++ were more energy efficient than both JavaScript and Python, in terms of the energy required to complete the benchmark.

RQ2: Is the faster programming language always the most energy efficient?

As seen in figure 4, the programming languages were ranked based on the amount of time they took to run the benchmark. The languages appeared in the same order as their energy consumption, with C taking the least time, followed by C++, JavaScript, and Python taking the most time. Therefore, it was observed that out of the programming languages tested, the faster the language, the more energy efficient it was in terms of energy used to perform the benchmark, resulting in less time spent running the benchmark and less overall energy consumption.

However, in terms of the rate of energy consumption, or watts, the trend was observed to differ. The interpreted languages were found to be more energy efficient than the compiled languages, with JavaScript being the most efficient at 26.33 watts, followed by Python at 37.63 watts, and C++ and C at 39.09 watts and 41.61 watts, respectively.

RQ3: How can the choice of programming language lower the environmental impact of big data processing and data mining?

As modern data processing accounts for a substantial amount of energy consumption and energy-related green house gas emissions, particularly as its field continues to grow, optimizing energy efficiency in processing by using the most energy efficient programming language for a specific task, where applicable, would have a positive global environmental impact. From the results of the lab, two scenarios can be identified for which the choice of programming language can lower the environmental impact of large-scale data processing.

The first scenario is when data centers have code that run to process a defined amount of data and terminates after the data has been processed. Examples include the training process of artificial intelligence and the validation process of validator nodes on blockchain networks, as both processes process a defined amount of data before terminating. For this scenario, the most energy efficient programming language out of the four tested would be C, as it required the least amount of energy to complete the benchmark.

The second scenario is when data centers have code that run around-the-clock, or continuously, regardless of input data. When running continuous operations, time consumption is usually not be a priority as code must continue running regardless of if there is data to process. Out of the four programming languages tested, JavaScript was found to be the most watt-efficient programming language. This means that it can do the same continuous processing as a less watt-efficient programming language, like C, but with less energy. By using the most watt-efficient programming language in situations where data centers are both run continuously and limited by wattage, such as when powered by a local renewable energy source, they would be able to support more simultaneous instances of a code without having to increase their energy consumption by methods such as purchasing energy from non-renewable energy sources. Therefore, watt-efficiency is especially important for continuous large-scale data processing and data mining as it impacts the feasibility and environmental impact of these operations.

In both scenarios, the selection of a programming language can be viewed as a supportive strategy of green coding, aimed at enhancing energy efficiency. It would contribute to achieving SDG 9 by retrofitting the data processing industry with the adoption of new environmentally responsible practices, especially if implemented as a new industry standard for improving energy efficiency. Through picking the most energy efficient programming language in each scenario, the energy efficiency of large-scale data processing would be better as compared to a suboptimal choice, thereby supporting the aspirations of SDG 7, particularly its subgoal SDG 7.3. Additionally, the improvement in energy efficiency within large-scale data processing would mean a potential reduction in energy-related green house gases, thus supporting SDG 13.

6. Conclusion

This research study investigated the impact of programming language on energy consumption in the processing of large data. The findings showed that the energy consumption of processing tasks differed across programming languages, with the compiled languages C and C++ being more energy-efficient, in terms of energy required per computation, than the interpreted languages JavaScript and Python. It was also observed that the faster the programming language was to complete the benchmark, the more energy-efficient it was. For code that terminate after the processing of defined data sets, the fastest language was optimal. However, when considering wattage, interpreted languages were more efficient than compiled languages, with JavaScript being the most efficient and the preferred language for around-the-clock operations. From an environment perspective, the selection of the most energy efficient programming language for each situation can significantly lower the environmental impact of big data processing and data mining, consequently supporting the achievement of the SDGs 7, 9 and 13.

Works Cited

Amodei, D., & Hernandez, D. (2018, May 16). *AI and compute*. Retrieved from OpenAI: https://openai.com/research/ai-and-compute

Eck., D. J. (2014, January 4). *Javanotes 6.0 -- Title Page*. Retrieved from HWS Math and CS: https://math.hws.edu/eck/cs124/javanotes6/c1/s1.html

Fanouraki, E. (2022, November 22). *SlashData*. Retrieved from SlashData : https://www.slashdata.co/blog/state-of-the-developer-nation-23rd-edition-the-fall-of-web-frameworks-coding-languages-blockchain-and-more

Fourment, M., & Gillings, M. (2008, February 05). *A comparison of common programming languages used in bioinformatics - BMC Bioinformatics*. Retrieved from BioMed Central: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-82

IBM. (2023, January 09). *Why green coding is a powerful catalyst for sustainability initiatives*. Retrieved from IBM: https://www.ibm.com/cloud/blog/green-coding

IEA. (2022, September). *Data Centres and data transmission networks – analysis*. Retrieved from IEA: https://www.iea.org/reports/data-centres-and-data-transmission-networks

Intel. (2022, August 2). *Running Average Power Limit Energy Reporting CVE-2020-8694,...* Retrieved from Intel: https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html

Portland State University. (n.d.). *How do programming languages make computers work? | STEMRobotics*. Retrieved from STEMRobotics: https://stemrobotics.cs.pdx.edu/node/4208.html

Rowell, E. (2016, August 23). *Big-O Algorithm Complexity Cheat Sheet (Know Thy Complexities!) @ericdrowell*. Retrieved from Big-O Cheat Sheet: https://www.bigocheatsheet.com/

United Nations. (n.d.). *Take Action for the Sustainable Development Goals - United Nations Sustainable Development*. Retrieved January 3, 2023, from United Nations: https://www.un.org/sustainabledevelopment/sustainable-development-goals/